CUBIT Developer’s Installation Instructions
by Tim Tautges
Revised April 9, 1997

Getting Started

Before getting started as a CUBIT developer, you will need the following things:

- An account on the LAN of Enchantment on Sandia’s Internal Restricted Network; you will need to be added to the ’cubit’ group on this LAN
- General knowledge of code development practices on UNIX machines

Gaining Access to the CUBIT Souce Code

To obtain CUBIT source code:

1. Log on to the IRN machine at Sandia (presently this resides on host sahp046.jal.sandia.gov). Initialize the CVSROOT environment variable as /usr/local/eng_sci/struct/CVS; if you are using csh, this can be accomplished using the command
   ```
   setenv CVSROOT /usr/local/eng_sci/struct/CVS
   ```
2. Make a subdirectory named ‘cubit’.
3. Obtain permission to check out CUBIT source code. This is usually accomplished by having the LAN administrator add you to the ‘cubit’ group, then upon login entering the command: **newgrp cubit**
4. In that directory ($HOME/cubit), execute the command:
   ```
   cvs checkout cubit
   ```
   This will create a subdirectory which contains all the CUBIT source files, and information necessary to check code in to the cvs code repository. The subdirectory will be given the name ‘cubit’, which can be changed to anything desired (most developers at Sandia use ‘source’, so that the cubit source resides in $HOME/cubit/source).

Setting Up a CUBIT Developer Environment

In addition to the CUBIT source code, you will need to locate a set of libraries and include files in order to compile CUBIT. You will also need to edit a configuration file to specify the locations of these files.
The following libraries and include files are needed to compile CUBIT:

- ACIS
  ACIS is a commercial solid modeling engine used by the CUBIT code. Typically, developers are either compiling on the LAN of Enchantment, or already have access to the ACIS library and include files. On the LAN of Enchantment, ACIS libraries and include files for the HP platform are located in $ES_HOME/cubit/acis/acis-[version], where [version] is currently ‘1.7’. Contact the CUBIT development team if you need Sun Solaris or SGI Irix versions of these libraries.
  ACIS development licenses must be purchased from Spatial Technologies; they can be reached at (303) 449-0649.

- HOOPS
  HOOPS is a commercial graphics package used by CUBIT for 3D graphics and rendering. CUBIT developers are covered under the HOOPS development license purchased from Ithaca software. On the LAN of Enchantment, HOOPS library and include files are located in $CUBIT/lib and $CUBIT/include (HP platform), or in $CUBIT/hoops (SGI, Sun). On all platforms, HOOPS libraries and includes are also available under $CUBIT/hoops/hoops-[version], where [version] is currently 'hoops-4.10'.

- LP Solve
  LP Solve is a commercial solver licensed by CUBIT; this library performs mixed integer linear programming solutions. LP Solve consists of one include file and one library file; on the LAN of Enchantment, the locations of these files are:
  $CUBIT/include/lpkit.h
  $CUBIT/lib/liblpk.a

- EXODUS II
  EXODUS II is a database format and code library used to write mesh data from CUBIT. This library consists of one include file and two UNIX archive libraries; these files are located in:
  $CUBIT/include/exodusII.h
  $CUBIT/lib/libexoIIv2c.a
  $CUBIT/lib/libnetcdf.a

- CCmakedepend
  CCmakedepend is used in the compilation process but isn't usually included in standard compiler installations. This program can be copied from $CUBIT/bin on the LAN of Enchantment or downloaded from the GNU project.

For those setting up remote development environments, these library and include files must be copied onto your system. Note that restrictions apply to some of this software. Overall, it is safe to assume that these libraries cannot be used for purposes other than compiling CUBIT in connection with your work as part of the CUBIT developer team. Under no circumstances are these files to be copied to a system other than the one on which you are developing CUBIT. Contact CUBIT developers for detailed restrictions for each package.

Environment

After obtaining the source code, developers in remote environments should tar up the entire source directory and move it to their development platform. It is very important that the entire directory be read into tar, since there are subdirectories that must also be copied.
The CUBIT make procedure requires the user to set up an auxilliary file which contains platform- and location-specific definitions. Templates for these files are located in $CUBIT_SOURCE/OtherFiles/CUBIT.xx, where xx is HP (HP version), SG (SGI version), or SS (Sun Solaris version). This file should be copied to $HOME/.cubit.yy (where yy is the lower case version of xx above).
Several definitions in .cubit.yy will have to be modified for your system. In particular, the following variables should be checked:

- X11_LIB_DIR, X11_INCLUDE (X Windows)
- EXODUS_LIB_DIR, EXODUS_INCLUDE (Exodus library, includes)
- LP_SOLVE_LIB_DIR, LP_SOLVE_INCLUDE (LP Solve package)
- SOURCE_DIR (User's CUBIT source directory, usually $HOME/cubit/source)
- HOOPS_LIB_DIR, HOOPS_INCLUDE (HOOPS rendering package)
- ACIS_DIR, AG_DIR (ACIS solid modeling engine)

The .cubit.yy file also contains definitions of the C++ compiler command and several others needed for building CUBIT; you should ensure that these tools are in your path or are referenced with a path name in .cubit.yy.

Compiling CUBIT

After moving the CUBIT source code onto the development platform, the following two commands will compile CUBIT:

```
make depend
make cubit
```

The first command builds a list of dependencies, which reduces compilation times for later compilations.
After the entire code has been built, the developer (you) can modify files locally and compile them into the CUBIT code using make.

Code Modification

Coding in CUBIT is governed by the CUBIT style guide, available from the CUBIT development team upon request. This style guide describes file, code, and comment naming conventions. All coding added to CUBIT should follow the conventions described in this document.
Maintaining a central code repository with developers modifying code in remote locations is filled with pitfalls. It is very important that the following procedure is followed exactly for committing changes to the CUBIT repository. Remember, changes committed to the repository by one developer affect all other developers upon their next update.
Modifications to CUBIT can be inserted into the repository as follows.

1. Transfer the ENTIRE CUBIT SOURCE DIRECTORY to the Sandia IRN machine using the tar utility. It is VERY important that the entire directory be moved, using tar, since cvs version control information is kept in a subdirectory of the source directory.
2. Perform a 'cvs update' in the CUBIT source directory which contains the modified files. Look at the output from cvs; if any files conflict with repository copies, these conflicts will be indicated in the source file with '<<<' and '>>>' characters. These conflicts must be resolved before the modified code can be checked in to the repository. CVS operations can also be done through the pcl-cvs package in emacs - contact the Sandia CUBIT development team for more information on this utility. See the man page (cvs(1)) and the cvs frequently asked questions (/usr/local/FAQ/cvs.FAQ on sahp046) for more information on cvs.
3. Compile CUBIT. ALWAYS compile before checking in changes to make sure the repository code will compile.

4. Execute the command ‘cvs commit source1.cc source2.hpp ...’, naming all the modified files on the command line. cvs will prompt for a log message for the commit, and then will commit changes to the repository.